Actions and Planning

➤ Agents reason in time

➤ Agents reason about time

Time passes as an agent acts and reasons.

Given a goal, it is useful for an agent to think about what it will do in the future to determine what it will do now.

Representing Time

Time can be modeled in a number of ways:

Discrete time Time can be modeled as jumping from one time point to another.

Continuous time You can model time as being dense.

Event-based time Time steps don't have to be uniform; you can consider the time steps between interesting events.

State space Instead of considering time explicitly, you can consider actions as mapping from one state to another.

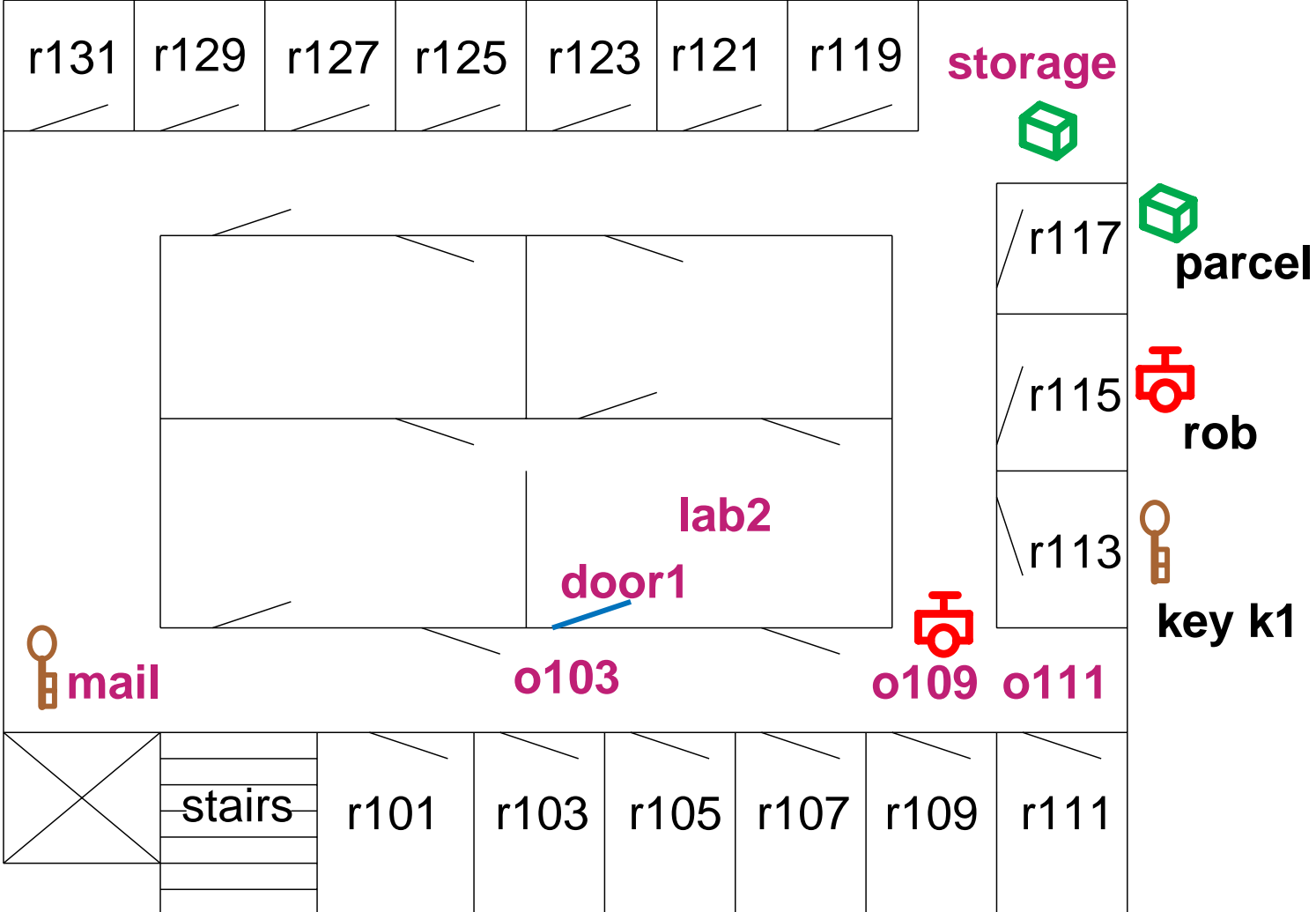You can model time in terms of **points** or **intervals.**

Time and Relations

When modeling relations, you distinguish two basic types:

➤ **Static relations** are those relations whose value does not depend on time.

➤ **Dynamic relations** are relations whose truth values depends on time. Either

   ➤ **derived relations** whose definition can be derived from other relations for each time,

   ➤ **primitive relations** whose truth value can be determined by considering previous times.

The Delivery Robot World

r131 r129 r127 r125 r123 r121 r119

storage

r117

parcel

r115

rob

lab2

r113

door1

key k1

mail

o103

o109 **o111**

stairs r101 r103 r105 r107 r109 r111

Modeling the Delivery Robot World

Individuals: rooms, doors, keys, parcels, and the robot.

Actions:

- ➤ move from room to room
- ➤ pick up and put down keys and packages
- ➤ unlock doors (with the appropriate keys)

Relations: represent

- ➤ the robot's position
- ➤ the position of packages and keys and locked doors
- ➤ what the robot is holding

Example Relations

➤ *at(Obj, Loc)* is true in a situation if object *Obj* is at location *Loc* in the situation.

➤ *carrying(Ag, Obj)* is true in a situation if agent *Ag* is carrying *Obj* in that situation.

➤ *sitting_at(Obj, Loc)* is true in a situation if object *Obj* is sitting on the ground (not being carried) at location *Loc* in the situation.

➤ *unlocked(Door)* is true in a situation if door *Door* is unlocked in the situation.

➤ *autonomous(Ag)* is true if agent *Ag* can move

autonomously. This is static.

➤ $opens(Key, Door)$ is true if key *Key* opens door *Door*. This is static.

➤ $adjacent(Pos_1, Pos_2)$ is true if position Pos_1 is adjacent to position Pos_2 so that the robot can move from Pos_1 to Pos_2 in one step.

➤ $between(Door, Pos_1, Pos_2)$ is true if *Door* is between position Pos_1 and position Pos_2. If the door is unlocked, the two positions are adjacent.

Actions

➤ *move*(*Ag*, *From*, *To*): agent *Ag* moves from location *From* to adjacent location *To*. The agent must be sitting at location *From*.

➤ *pickup*(*Ag*, *Obj*) agent *Ag* picks up *Obj*. The agent must be at the location that *Obj* is sitting.

➤ *putdown*(*Ag*, *Obj*) the agent *Ag* puts down *Obj*. It must be holding *Obj*.

➤ *unlock*(*Ag*, *Door*) agent *Ag* unlocks *Door*. It must be outside the door and carrying the key to the door.

Initial Situation

sitting_at(*rob*, *o*109).

sitting_at(*parcel*, *storage*).

sitting_at(*k*1, *mail*).

Static Facts

between(*door*1, *o*103, *lab*2).

opens(*k*1, *door*1).

autonomous(*rob*).

Derived Relations

$at(Obj, Pos) \leftarrow sitting\_at(Obj, Pos)$.

$at(Obj, Pos) \leftarrow carrying(Ag, Obj) \wedge at(Ag, Pos)$.

$adjacent(o109, o103)$.

$adjacent(o103, o109)$.

. . .

$adjacent(lab2, o109)$.

$adjacent(P_1, P_2) \leftarrow$

$\qquad between(Door, P_1, P_2) \wedge$

$\qquad unlocked(Door)$.